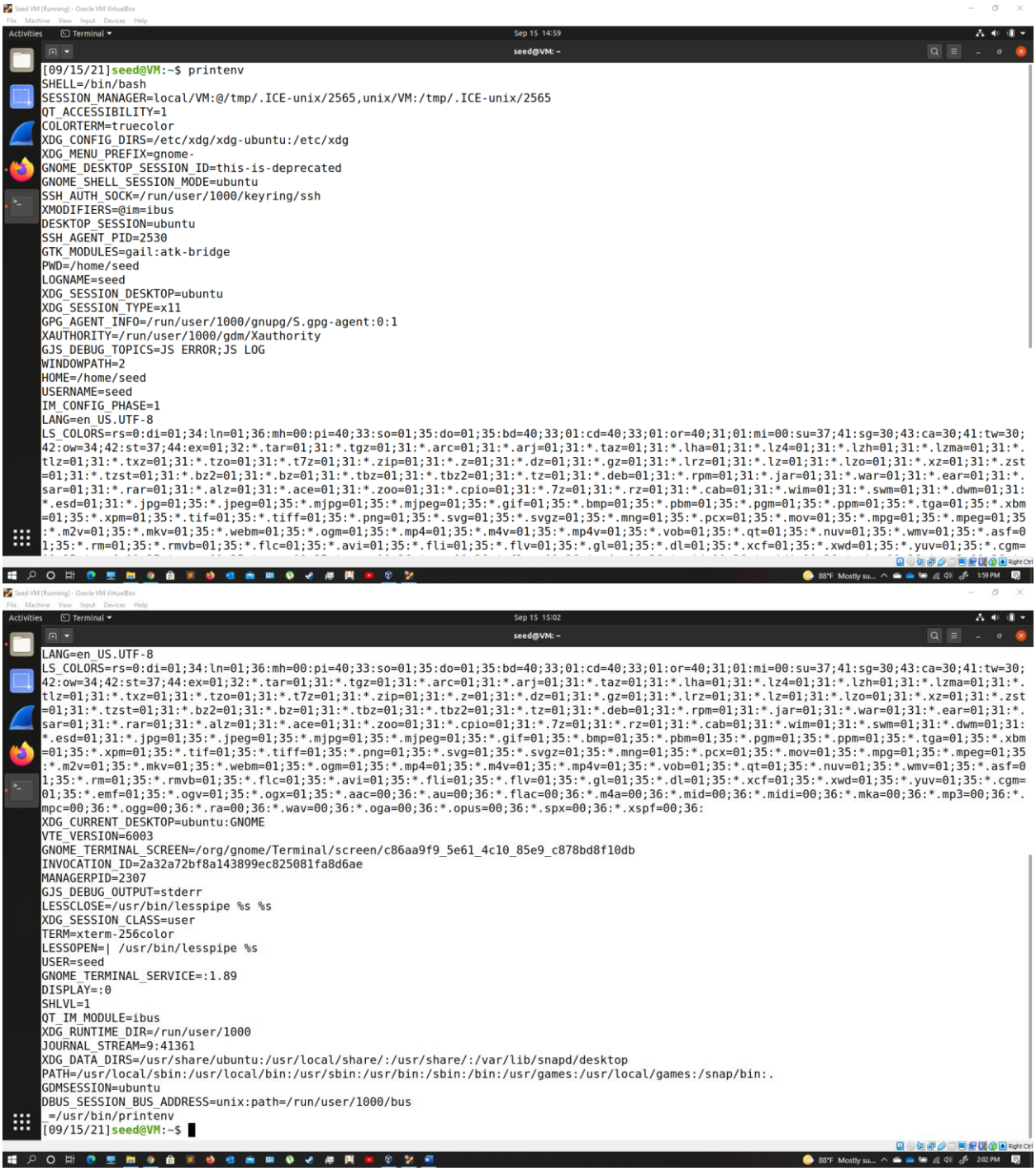
Secure programing assignment 1

CSE 5382-001

Submitted by:
Name: Anuraag Venkatapuram Sreenivas
Student id: 1001716458

Task 1: Manipulating Environment Variables

Use “printenv” or “env” command to print out the environment variables:



Observation: Prints all the environment variables

Observation: Displays PWD environment variable

Use “export” and “unset” to set or unset environment variables:

```
[09/15/21]seed@VM:~$ export new_temp_env="/home/seed"
[09/15/21]seed@VM:~$ printenv new_temp_env
/home/seed
```

Observation: Creating an environment variable using “export” and using “printenv” command to see if the variable is created at that location.

```
[09/15/21]seed@VM:~$ unset new_temp_env
[09/15/21]seed@VM:~$ printenv new_temp_env
[09/15/21]seed@VM:~$
```

Observation: using “unset” to remove the variable from the list.

Task 2: Passing Environment Variables from Parent Process to Child Process

Step1:

```
[09/15/21]seed@VM:~$ cd Labsetup/
[09/15/21]seed@VM:~/Labsetup$ gcc myprintenv.c
[09/15/21]seed@VM:~/Labsetup$ a.out > file
[09/15/21]seed@VM:~/Labsetup$ cat myprintenv.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void printenv()
{
  int i = 0;
  while (environ[i] != NULL) {
     printf("%s\n", environ[i]);
     i++;
  }
}

void main()
{
  pid_t childPid;
  switch(childPid = fork()) {
    case 0:  /* child process */
      printenv();
      exit(0);
    default:  /* parent process */
      // printenv();
      exit(0);
  }
}
[09/15/21]seed@VM:~/Labsetup$
```

Observation: executing child process

Step2:

```
[09/15/21]seed@VM:~/Labsetup$ vi myprintenv.c
[09/15/21]seed@VM:~/Labsetup$ cat myprintenv.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void printenv()
{
  int i = 0;
  while (environ[i] != NULL) {
     printf("%s\n", environ[i]);
     i++;
  }
}

void main()
{
  pid_t childPid;
  switch(childPid = fork()) {
    case 0:  /* child process */
      // printenv();
      exit(0);
    default:  /* parent process */
      printenv();
      exit(0);
  }
}
[09/15/21]seed@VM:~/Labsetup$ gcc myprintenv.c
[09/15/21]seed@VM:~/Labsetup$ a.out > file2
[09/15/21]seed@VM:~/Labsetup$
```

Observation: executing parent process

Step3:

```
[09/15/21]seed@VM:~/Labsetup$ diff file file2
[09/15/21]seed@VM:~/Labsetup$
```

Observation: there is no difference in the output files created in the either step's as the "diff" command doesn't give out any output. (can be verified using cat command on each file) Hence it can be interpreted that the child process inherits all the environment variables from parent.

Task 3: Environment Variables and execve()

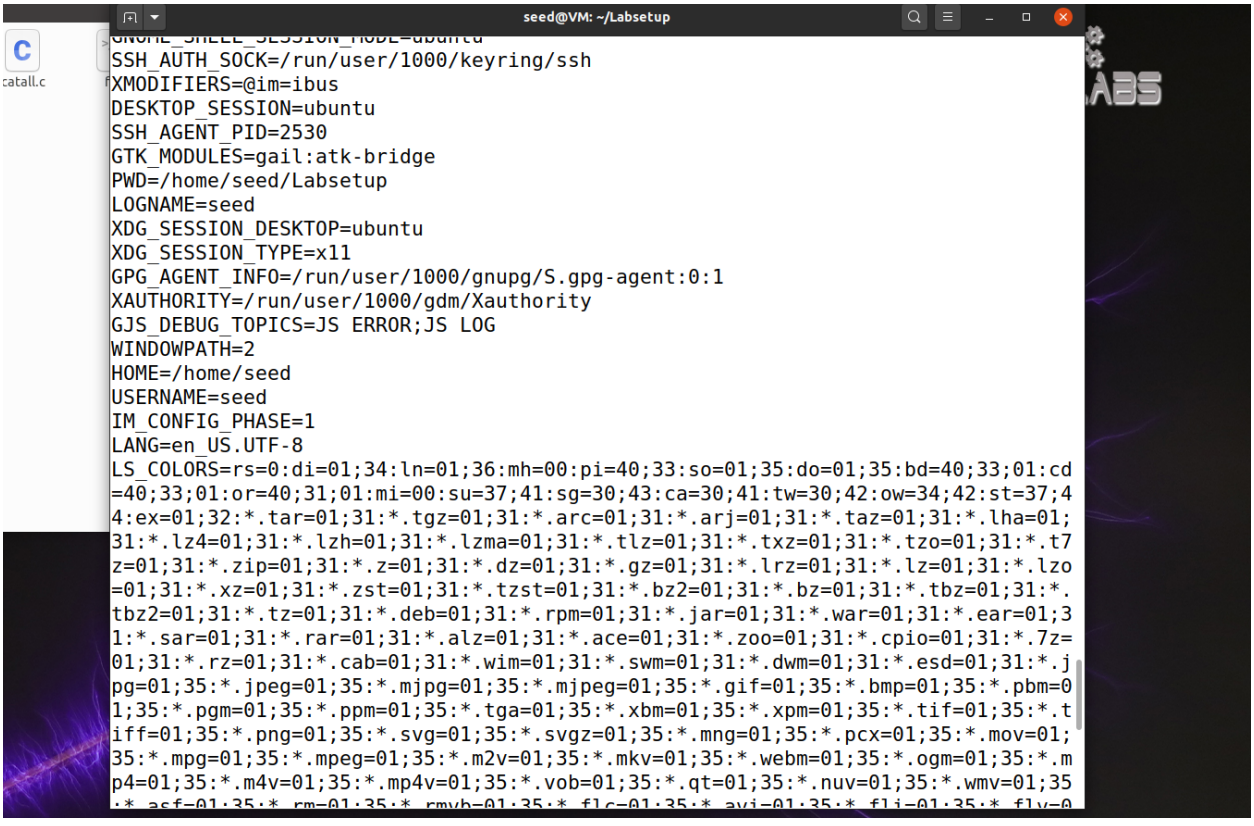
Step1:

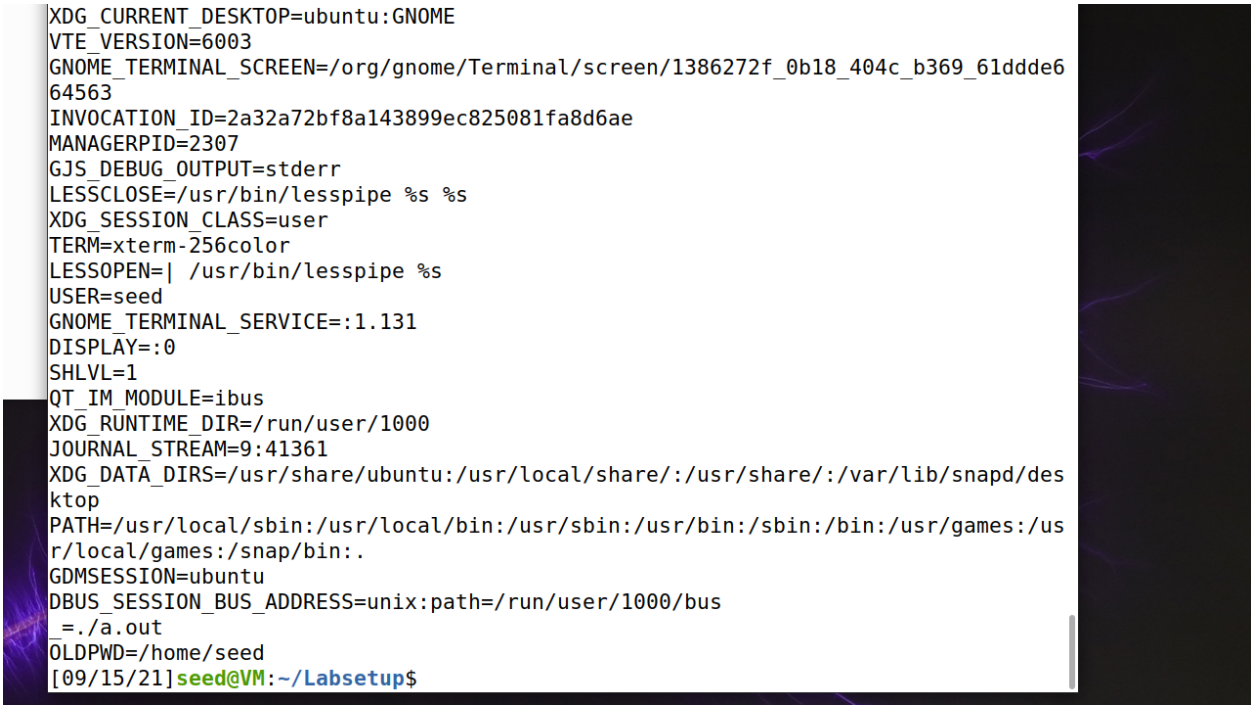
```
[09/15/21]seed@VM:~/Labsetup$ cat myenv.c
#include <unistd.h>

extern char **environ;

int main()
{
  char *argv[2];

  argv[0] = "/usr/bin/env";
  argv[1] = NULL;

  execve("/usr/bin/env", argv, NULL);

  return 0 ;
}

[09/15/21]seed@VM:~/Labsetup$ gcc myenv.c
[09/15/21]seed@VM:~/Labsetup$ a.out > task3_1
[09/15/21]seed@VM:~/Labsetup$ cat task3_1
[09/15/21]seed@VM:~/Labsetup$
```

Observation: simply executing “/usr/bin/env” but we do not get any environment variables as NULL is used as the third parameter

Step2:

```
[09/15/21]seed@VM:~/Labsetup$ vi myenv.c
[09/15/21]seed@VM:~/Labsetup$ cat myenv.c
#include <unistd.h>

extern char **environ;

int main()
{
  char *argv[2];

  argv[0] = "/usr/bin/env";
  argv[1] = NULL;

  execve("/usr/bin/env", argv, environ);

  return 0 ;
}

[09/15/21]seed@VM:~/Labsetup$ gcc myenv.c
[09/15/21]seed@VM:~/Labsetup$ a.out > task3_2
[09/15/21]seed@VM:~/Labsetup$ cat task3_2
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2565,unix/VM:/tmp/.ICE-unix/2565
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
```





Observation: after changing the 3rd argument from null to environ and compiling and executing the same program, we can see that the process prints all the environment variables that it inherits. Here we are passing the environment variables in while invoking the execve() method along with “/usr/bin/env”.

Task 4: Environment Variables and system()

```
[09/15/21]seed@VM:~/Labsetup$ cat task4.c
#include <stdio.h>
#include <stdlib.h>
int main()
{
        system("/usr/bin/env");
        return 0 ;
}
[09/15/21]seed@VM:~/Labsetup$ gcc task4.c
[09/15/21]seed@VM:~/Labsetup$ a.out
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=2530
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
OLDPWD=/home/seed
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
MANAGERPID=2307
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COLORTERM=truecolor
IM_CONFIG_PHASE=1
LOGNAME=seed
JOURNAL_STREAM=9:41361
_=./a.out
XDG_SESSION_CLASS=user
USERNAME=seed
TERM=xterm-256color
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
WINDOWPATH=2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
```

Observation: The program prints all the environment variables of the current process. The system() command would use fork() to create a child process which executes the shell command. Hence the child process would display all the environment variables.

Task 5: Environment Variable andSet-UIDPrograms:

Step1: running the given program

```
[09/15/21]seed@VM:~/Labsetup$ vi task5.c
[09/15/21]seed@VM:~/Labsetup$ cat task5.c
#include <stdio.h>
#include <stdlib.h>
extern char**environ;
int main()
{
        int i = 0;
        while (environ[i] != NULL)
        {
                printf("%s\n", environ[i]);
                i++;
        }
}
[09/15/21]seed@VM:~/Labsetup$
```

Step2: changing permissions

```
[09/15/21]seed@VM:~/Labsetup$ gcc task5.c -o task5
[09/15/21]seed@VM:~/Labsetup$ sudo chown root task5
[09/15/21]seed@VM:~/Labsetup$ sudo chmod 4755 task5
```

Step3:

```
[09/15/21]seed@VM:~/Labsetup$ export task5_env="task5env"
[09/15/21]seed@VM:~/Labsetup$ printenv task5_env
task5env
[09/15/21]seed@VM:~/Labsetup$ ./task5
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2565,unix/VM:/tmp/.ICE-unix/2565
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2530
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Labsetup
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
IM_CONFIG_PHASE=1
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
```

```
task5_env=task5env
```

Observation: the new environment variable (shown in second image) that was created was inherited along with the other environment variables and are displayed when the program is executed.

Task 6: The PATH Environment Variable and Set-UID Programs:

```
[09/15/21]seed@VM:~/Labsetup$ vi task6.c
[09/15/21]seed@VM:~/Labsetup$ cat task6.c
#include<stdlib.h>
int main()
{
        system("ls");
        return 0;
}
```

Compiling the above program, and change its owner to root, and make it a Set-UID program.

```
[09/15/21]seed@VM:~/Labsetup$ gcc task6.c -o task6
[09/15/21]seed@VM:~/Labsetup$ sudo chown root task6
[09/15/21]seed@VM:~/Labsetup$ sudo chmod 4755 task6
```

```
[09/15/21]seed@VM:~/Labsetup$ ./task6
a.out       catall.c  file2    myprintenv.c  task3_2  task5    task6
cap_leak.c  file      myenv.c  task3_1       task4.c  task5.c  task6.c
```

Changing the code to test our own command for syterm():

```
[09/17/21]seed@VM:~/Labsetup$ vi task6.c
[09/17/21]seed@VM:~/Labsetup$ cat task6.c
#include<stdlib.h>
int main()
{
        system("ifconfig");
        return 0;
}
[09/17/21]seed@VM:~/Labsetup$
```

```
[09/17/21]seed@VM:~/Labsetup$ gcc task6.c -o task6
[09/17/21]seed@VM:~/Labsetup$ sudo chown root task6
[09/17/21]seed@VM:~/Labsetup$ sudo chmod 4755 task6
[09/17/21]seed@VM:~/Labsetup$ ./task6
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:3e:eb:4e:d6  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.4  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::6a04:3638:16b4:83b  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:df:57:91  txqueuelen 1000  (Ethernet)
        RX packets 725  bytes 918577 (918.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 590  bytes 49856 (49.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 162  bytes 13130 (13.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 162  bytes 13130 (13.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[09/17/21]seed@VM:~/Labsetup$
```

Observation: the system() command invokes fork() which creates new process and executes the command. This execution could be affected by other users since system() command would refer to the environment variables. Therefore this prgm is proven to be very dangerous.

Task 7: The LD PRELOAD Environment Variable and Set-UID Programs:

Step1.1: building a dynamic link library

```
[09/15/21]seed@VM:~/Labsetup$ vi mylib.c
[09/15/21]seed@VM:~/Labsetup$ cat mylib.c
#include <stdio.h>
void sleep (int s)
{
        /*If this is invoked by a privileged program,you can do damages here!*/
        printf("I am not sleeping!\n");
}
[09/15/21]seed@VM:~/Labsetup$
```

Step1.2: compiling the program

```
[09/15/21]seed@VM:~/Labsetup$ gcc -fPIC -g -c mylib.c
[09/15/21]seed@VM:~/Labsetup$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[09/15/21]seed@VM:~/Labsetup$
```

Step1.3: set the LD PRELOAD environment variable

```
[09/15/21]seed@VM:~/Labsetup$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/15/21]seed@VM:~/Labsetup$ printenv LD_PRELOAD
./libmylib.so.1.0.1
```

Step1.4: compiling myprog.c

```
[09/15/21]seed@VM:~/Labsetup$ vi myprog.c
[09/15/21]seed@VM:~/Labsetup$ cat myprog.c
#include <unistd.h>
int main()
{
        sleep(1);
        return 0;
}
[09/15/21]seed@VM:~/Labsetup$ gcc myprog.c -o myprog
```

Step2.1: Make myprog regular program, and run it as a normal user.

```
[09/15/21]seed@VM:~/Labsetup$ ./myprog
I am not sleeping!
```

Observation: we see the custom sleep() method that we created is being invoked as we have included it in the environment variable

Step2.2: Make myprog a Set-UID root program, and run it as a normal user.

```
root@VM:/home/seed/Labsetup# gcc -o myprog myprog.c
root@VM:/home/seed/Labsetup# chmod u+s myprog
root@VM:/home/seed/Labsetup# exit
exit
[09/15/21]seed@VM:~/Labsetup$ ls -l
total 156
-rwxrwxr-x 1 seed seed 16696 Sep 15 17:51 a.out
-rw-rw-r-- 1 seed seed   761 Dec 27  2020 cap_leak.c
-rw-rw-r-- 1 seed seed   471 Feb 19  2021 catall.c
-rw-rw-r-- 1 seed seed  2953 Sep 15 15:28 file
-rw-rw-r-- 1 seed seed  2953 Sep 15 15:32 file2
-rwxrwxr-x 1 seed seed 18688 Sep 15 18:44 libmylib.so.1.0.1
-rw-rw-r-- 1 seed seed   183 Sep 15 17:23 myenv.c
-rw-rw-r-- 1 seed seed   148 Sep 15 18:41 mylib.c
-rw-rw-r-- 1 seed seed  5944 Sep 15 18:44 mylib.o
-rw-rw-r-- 1 seed seed   418 Sep 15 15:32 myprintenv.c
-rwsr-xr-x 1 root root 16696 Sep 15 19:11 myprog
-rw-rw-r-- 1 seed seed    57 Sep 15 18:49 myprog.c
-rw-rw-r-- 1 seed seed     0 Sep 15 17:16 task3_1
-rw-rw-r-- 1 seed seed  2953 Sep 15 17:23 task3_2
-rw-rw-r-- 1 seed seed    91 Sep 15 17:50 task4.c
-rwxrwxr-x 1 root seed 16768 Sep 15 18:03 task5
-rw-rw-r-- 1 seed seed   160 Sep 15 18:00 task5.c
-rwxrwxr-x 1 4755 seed 16696 Sep 15 18:24 task6
-rw-rw-r-- 1 seed seed    60 Sep 15 18:24 task6.c
[09/15/21]seed@VM:~/Labsetup$ ./myprog
[09/15/21]seed@VM:~/Labsetup$
```

Observation: when a program is set as a privileged root program it would ignore the previously set LD_PRELOAD variable while executing. Hence the custom sleep function is not called here.

Step2.3: Make myprog a Set-UID root program, export the LD PRELOAD environment variable again in the root account and run it.

```
root@VM:/home/seed/Labsetup# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/Labsetup# ./myprog
I am not sleeping!
root@VM:/home/seed/Labsetup#
```

Observation: since the LD_PRLOAD is set to the point to the custom library, again the privileged root program would call the custom Sleep() method.

Step2.4: Make myprog a Set-UID user1 program (i.e., the owner is user1, which is another user account), export the LD PRELOAD environment variable again in a different user's account (not-root user) and run it.

```
root@VM:/home/seed/Labsetup# adduser user1
Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
ERROR: ld.so: object './libmylib.so.1.0.1' from LD_PRELOAD cannot be preloaded (c
annot open shared object file): ignored.
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
        Full Name []: usre1
        Room Number []: 1
        Work Phone []: 1
        Home Phone []: 1
        Other []: 1
Is the information correct? [Y/n] y
root@VM:/home/seed/Labsetup# chown user1 myporg
chown: cannot access 'myporg': No such file or directory
root@VM:/home/seed/Labsetup# chown user1 myprog
```

```
[09/15/21]seed@VM:~/Labsetup$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/15/21]seed@VM:~/Labsetup$ ./myprog
I am not sleeping!
```

Observation: we see that out custom sleep() method is being invoked in this case.

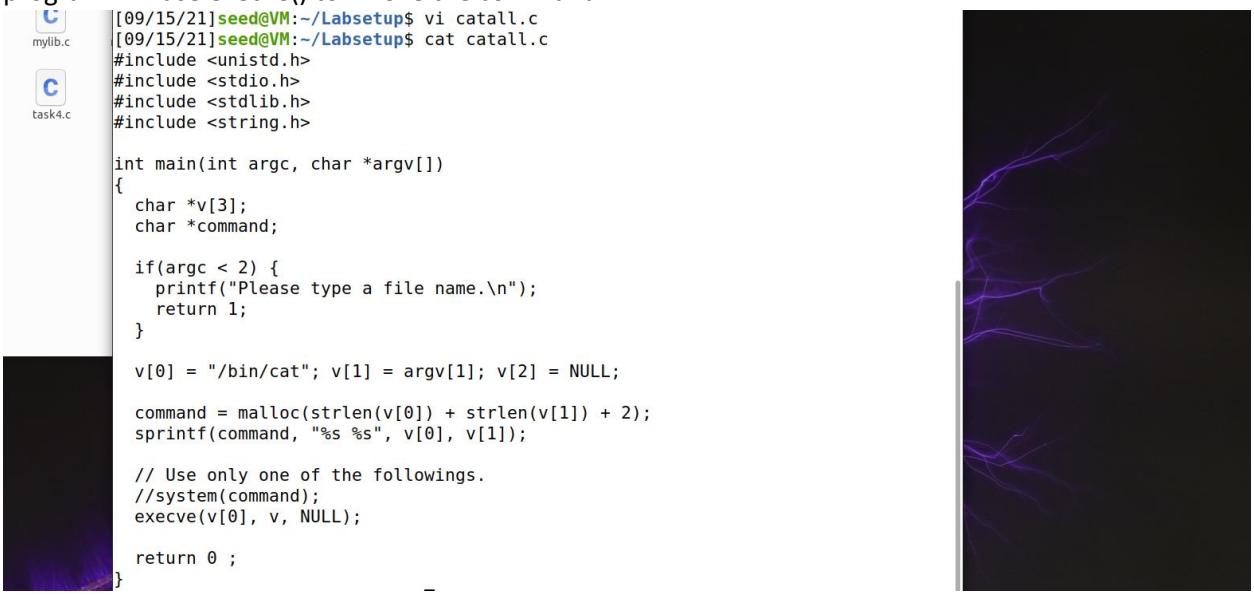
Task 8: Invoking External Programs Using system() versus execve()

Step1: Compile the above program, make it a root-owned Set-UID program. The program will use system() to invoke the command.

```
[09/15/21]seed@VM:~/Labsetup$ gcc catall.c -o catall
[09/15/21]seed@VM:~/Labsetup$ sudo chown root catall
[09/15/21]seed@VM:~/Labsetup$ sudo chmod 4755 catall
```

```
[09/15/21]seed@VM:~/Labsetup$ ./catall catall.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
  char *v[3];
  char *command;

  if(argc < 2) {
    printf("Please type a file name.\n");
    return 1;
  }

  v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;

  command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
  sprintf(command, "%s %s", v[0], v[1]);

  // Use only one of the followings.
  system(command);
  // execve(v[0], v, NULL);

  return 0 ;
}
[09/15/21]seed@VM:~/Labsetup$
```

Step 2: Comment out the system(command) statement, and uncomment the execve() statement; the program will use execve() to invoke the command.



```
[09/15/21]seed@VM:~/Labsetup$ gcc catall.c -o catall
[09/15/21]seed@VM:~/Labsetup$ sudo chown root catall
[09/15/21]seed@VM:~/Labsetup$ sudo chmod 4755 catall
[09/15/21]seed@VM:~/Labsetup$ ./catall catall.c
```

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
  char *v[3];
  char *command;

  if(argc < 2) {
    printf("Please type a file name.\n");
    return 1;
  }

  v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;

  command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
  sprintf(command, "%s %s", v[0], v[1]);

  // Use only one of the followings.
  //system(command);
  execve(v[0], v, NULL);

  return 0 ;
}
```

Observation: execve() uses root environment variables. And these variables can be set by privileged users only. Hence, using execve() does not cause any attack in step 1.

Task 9: Capability Leaking

```
[09/15/21]seed@VM:~/Labsetup$ cat cap_leak.c
```

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

void main()
{
  int fd;
  char *v[2];

  /* Assume that /etc/zzz is an important system file,
   * and it is owned by root with permission 0644.
   * Before running this program, you should create
   * the file /etc/zzz first. */
  fd = open("/etc/zzz", O_RDWR | O_APPEND);
  if (fd == -1) {
     printf("Cannot open /etc/zzz\n");
     exit(0);
  }

  // Print out the file descriptor value
  printf("fd is %d\n", fd);

  // Permanently disable the privilege by making the
  // effective uid the same as the real uid
  setuid(getuid());

  // Execute /bin/sh
  v[0] = "/bin/sh"; v[1] = 0;
  execve(v[0], v, 0);
}
```

```
[09/15/21]seed@VM:~/Labsetup$
[09/15/21]seed@VM:~/Labsetup$ gcc cap_leak.c  -o capleak
[09/15/21]seed@VM:~/Labsetup$ sudo chown root capleak
[09/15/21]seed@VM:~/Labsetup$ sudo chmod 4755 capleak
[09/15/21]seed@VM:~/Labsetup$ ./capleak
Cannot open /etc/zzz
```

Observation: non root users will not have write access to the file zzz. When the file owner is changed to root the current privileges are revoked making the file inaccessible.